영화 추천 엔진

[Import library]

```
import numpy as np
import pandas as pd
import json
```

[Load Dataset from kaggle]

```
meta = pd.read_csv('movies_metadata.csv')
#meta = pd.read_csv('movies_metadata.csv',low_memory=False)
meta.head()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3165: DtypeWarning: Columns (10)
have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_langua
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt0113497	en
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt0113228	en
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	31357	tt0114885	en
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}]	NaN	11862	tt0113041	en

5 rows × 24 columns

```
meta = meta[['id','original_title', 'original_language', 'genres']]
meta = meta.rename(columns={'id':'movieId'})
meta = meta[meta['original_language'] == 'en']
meta.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	movieId	original_title	original_language	genres
0	862	Toy Story	en	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...
1	8844	Jumanji	en	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...
2	15602	Grumpier Old Men	en	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...
3	31357	Waiting to Exhale	en	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...
4	11862	Father of the Bride Part II	en	[{'id': 35, 'name': 'Comedy'}]

```
ratings = pd.read_csv('ratings.csv')
ratings = ratings[['userId', 'movieId', 'rating']]
ratings.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	userId	movieId	rating
0	1	110	1.0
1	1	147	4.5
2	1	858	5.0
3	1	1221	5.0
4	1	1246	5.0

```
ratings.describe()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	userId	movieId	rating
count	2.602429e+07	2.602429e+07	2.602429e+07
mean	1.350371e+05	1.584911e+04	3.528090e+00
std	7.817620e+04	3.108526e+04	1.065443e+00
min	1.000000e+00	1.000000e+00	5.000000e-01
25%	6.716400e+04	1.073000e+03	3.000000e+00
50%	1.351630e+05	2.583000e+03	3.500000e+00
75%	2.026930e+05	6.503000e+03	4.000000e+00
max	2.708960e+05	1.762750e+05	5.000000e+00

[Refine Dataset]

```
#pandas.to_numeric : to_numeric은 데이터를 숫자 형식으로 바꿔주는 역할
#errors = 'ignore' : 숫자로 변경할 수 없는 데이터라면 숫자로 변경하지 않고 원본 데이터를 그대로 반환
#errors = 'coerce' : 숫자로 변경할 수 없는 데이터라면 기존데이터를 지우고 NaN으로 반환
#errors = 'raise' : 숫자로 변경할 수 없는 데이터라면 에러를 일으키며 코드를 중단
meta.movieId = pd.to_numeric(meta.movieId, errors='coerce')
ratings.movieId = pd.to_numeric(ratings.movieId, errors='coerce')
```

```
def parse_genres(genres_str):
    genres = json.loads(genres_str.replace('\'','"'))

    genres_list = []
    for g in genres:
        genres_list.append(g['name'])

    return genres_list

meta['genres'] = meta['genres'].apply(parse_genres)

meta.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	movieId	original_title	original_language	genres
0	862	Toy Story	en	[Animation, Comedy, Family]
1	8844	Jumanji	en	[Adventure, Fantasy, Family]
2	15602	Grumpier Old Men	en	[Romance, Comedy]
3	31357	Waiting to Exhale	en	[Comedy, Drama, Romance]
4	11862	Father of the Bride Part II	en	[Comedy]

[Merge Meta and Ratings]

```
#how = 'inner' : Inner join (#left, right,inner(default),outer)
#on = 'X '': merger의 기준이 되는 key변수


data = pd.merge(ratings, meta, on='movieId', how='inner')

data.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	userId	movieId	rating	original_title	original_language	genres
0	1	858	5.0	Sleepless in Seattle	en	[Comedy, Drama, Romance]
1	3	858	4.0	Sleepless in Seattle	en	[Comedy, Drama, Romance]
2	5	858	5.0	Sleepless in Seattle	en	[Comedy, Drama, Romance]
3	12	858	4.0	Sleepless in Seattle	en	[Comedy, Drama, Romance]
4	20	858	4.5	Sleepless in Seattle	en	[Comedy, Drama, Romance]

[Pivot Table]

```
matrix = data.pivot_table(index = 'userId', columns = 'original_title', values='rating')
matrix.head(20)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

original_title	!Women Art Revolution	$5 a Day	'Gator Bait	'R Xmas	'Twas the Night Before Christmas	(A)Sexual	...And the Pursuit of Happiness	10 Items or Less	10 Things I Hate About You	10,000 BC	...	Мой сводный брат Франкенштейн
userId												
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
12	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
13	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
15	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
18	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
19	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN

20 rows × 5298 columns

Pearson Correlation coefficient

- 상관계수(correlation coefficient)란 두 변수가 어떤 상관 관계를 가지는지를 의미하는 수치이다.
- **피어슨 상관 계수**는 다양한 상황에서 쓰이지만, normalized된 cosine similarity를 계산하는 것이기 때문에 **similarity**로도 해석할 수 있다.
- similarity로 쓰이는 예로는 **추천 시스템**이 있다.

```
GENRE_WEIGHT = 0.1

def pearsonR(s1, s2):
    s1_c = s1 - s1.mean()
    s2_c = s2 - s2.mean()
    return np.sum(s1_c * s2_c) / np.sqrt(np.sum(s1_c**2)*np.sum(s2_c**2))

#input으로 title, pivot table을 받고, n개의 영화를 추천받는다. similar_genres에 가중치를 둔다.
def recommend(input_movie, matrix, n, similar_genre=True):
    input_genres = meta[meta['original_title']==input_movie]['genres'].iloc(0)[0]

    result = []
    for title in matrix.columns:
        if title == input_movie:
            #똑같은 영화를 추천받지 않기 위해 건너뜀
            continue

        #rating comparison
        cor = pearsonR(matrix[input_movie], matrix[title])

        #genre comparison
        if similar_genre and len(input_genres)>0:
            temp_genres = meta[meta['original_title']==input_movie]['genres'].iloc(0)[0]

            #np.isin() : 배열을 비교하여 똑같은 요소가 있으면 True를 반환
            same_count = np.sum(np.isin(input_genres, temp_genres))
            cor += (GENRE_WEIGHT * same_count)



        if np.isnan(cor):
            continue
        else:
            result.append((title, '{:.2f}'.format(cor),temp_genres))
    result.sort(key = lambda r: r[1],reverse=True)
    #rating이 높은 순서대로 내림차순 정렬

    return result[:n]
```

[Prediction]

```
recommend_result = recommend('The Dark Knight', matrix, 10, similar_genre=True)

pd.DataFrame(recommend_result, columns = ['Title', 'Correlation', ' Genre'])
```

```
<ipython-input-31-ccfb8e487443>:6: RuntimeWarning: invalid value encountered in double_scalars
  return np.sum(s1_c * s2_c) / np.sqrt(np.sum(s1_c**2)*np.sum(s2_c**2))
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Title	Correlation	Genre
0	The Acid Eaters	0.48	[Drama, Action, Crime, Thriller]
1	Amadeus	0.47	[Drama, Action, Crime, Thriller]
2	American Milkshake	0.47	[Drama, Action, Crime, Thriller]
3	Beach Party	0.47	[Drama, Action, Crime, Thriller]
4	Bride of Frankenstein	0.47	[Drama, Action, Crime, Thriller]
5	Dead Man Walking	0.47	[Drama, Action, Crime, Thriller]
6	Hail Caesar	0.47	[Drama, Action, Crime, Thriller]
7	Hotel Rwanda	0.47	[Drama, Action, Crime, Thriller]
8	Intern Academy	0.47	[Drama, Action, Crime, Thriller]
9	King Kong	0.47	[Drama, Action, Crime, Thriller]

```
recommend_result = recommend('The Acid Eaters', matrix, 10, similar_genre=True)

pd.DataFrame(recommend_result, columns = ['Title', 'Correlation', ' Genre'])
```

```
<ipython-input-31-ccfb8e487443>:6: RuntimeWarning: invalid value encountered in double_scalars
  return np.sum(s1_c * s2_c) / np.sqrt(np.sum(s1_c**2)*np.sum(s2_c**2))
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Title	Correlation	Genre
0	Taste the Waste	0.94	[Drama]
1	Tomorrowland	0.91	[Drama]
2	Graffiti Bridge	0.90	[Drama]
3	The Silver Screen: Color Me Lavender	0.89	[Drama]
4	Hail Caesar	0.88	[Drama]
5	The Falls	0.88	[Drama]
6	Intern Academy	0.87	[Drama]
7	La edad del amor	0.87	[Drama]
8	Metsän Tarina	0.87	[Drama]
9	The House Across the Street	0.87	[Drama]